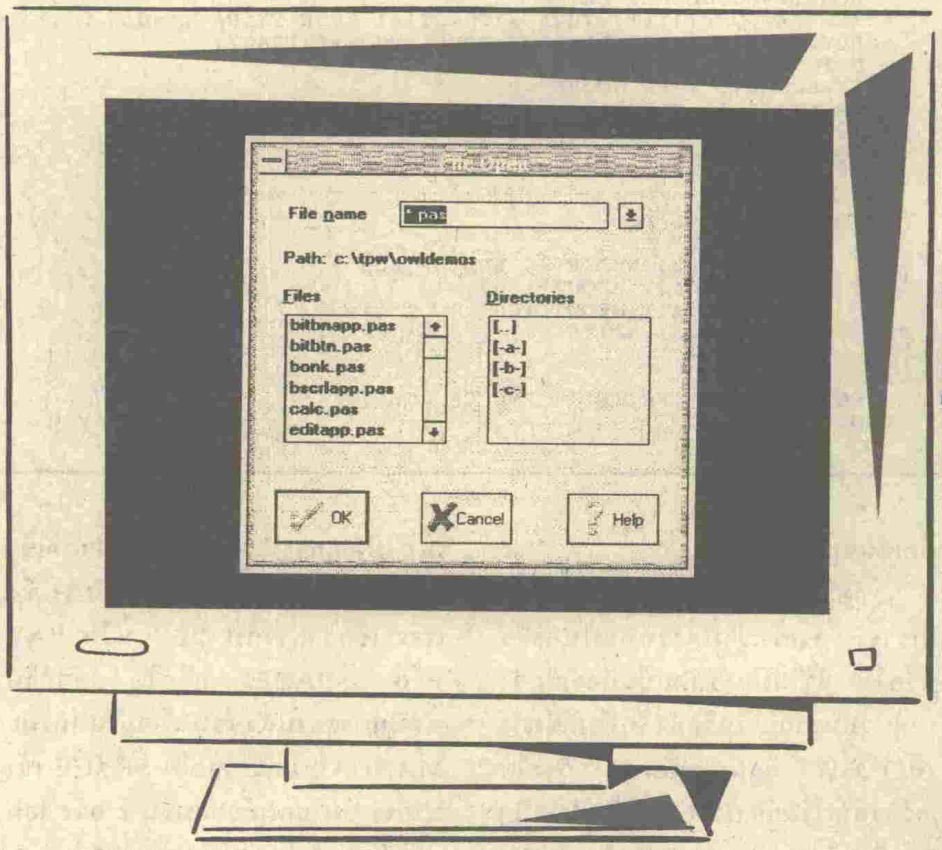
บุญเลิศ เอี่ยมทัศนา

เทอร์โบปาสคาล
กับวินโดวส์ : เตรียมการ (2)

วิธีการติดตั้ง เทอร์โบปาสคาล กับ วินโดวส์ 3.0 และการเขียนโปรแกรมในแบบโอโอพี



การเขียนโปรแกรมบนวินโดวส์ด้วยเทอร์โบปาสคาล จำเป็นต้องทราบเรื่องราวของวินโดวส์ ภาษาปาสคาล และการเขียนโปรแกรมในแบบโอโอพี ซึ่งได้กล่าวถึงเรื่องราวของวินโดวส์ไปแล้วพอสมควร ส่วนเรื่องของภาษาปาสคาลต้องขอให้ทราบมาก่อนบ้างแล้ว ในฉบับนี้จึงจะกล่าวถึงการเขียนโปรแกรมในแบบโอโอพีอย่างกว้าง ๆ แต่สิ่งที่ไม่ควรมองข้ามคือ การใช้งานเทอร์โบปาสคาล จึงได้นำมากล่าวด้วยตั้งแต่การติดตั้ง

การติดตั้ง

การติดตั้งเทอร์โบปาสคาลคงดำเนินการตามหนังสือคู่มือ คือ หลังจากการติดตั้งวินโดวส์แล้วให้ติดตั้งเทอร์โบปาสคาล โดยใส่ดิสค์ที่มีไฟล์ install.exe ที่ไดรฟ์ A แล้วใช้คำสั่งดอสว่า

```
C:>win a: install
```

จะเป็นการรันวินโดวส์ก่อนแล้วจึงรันโปรแกรม install ของเทอร์โบ-ปาสคาล จากนี้ก็เพียงแต่เปลี่ยนแผ่นดิสค์ตามคำแนะนำบนจอภาพ เมื่อติดตั้งเสร็จแล้วให้เพิ่มเติม PATH ในไฟล์ AUTOEXEC.BAT ให้มี

```
C:\TPW;C\TPW\UTILS
```

การรัน

การรันเทอร์โบปาสคาลทำได้ 2 แบบ คือ

ก. รันพร้อมกับวินโดวส์ คือ

```
C:>win tpw
```

ข. หลังจากรันวินโดวส์แล้ว

1. เปิดใช้งาน Turbo Pascal (เป็นหน้าต่างกลุ่ม)

2. เปิดใช้งาน TPW (เป็นหน้าต่างงาน)

จะได้ภาพบนจอภาพดังรูปที่ (ซึ่งถ้ารันพร้อมกับวินโดวส์ Program Manager จะอยู่ในรูปของไอคอน) จะเห็นว่ามีหน้าต่างที่ใช้ชื่อ Turbo Pascal

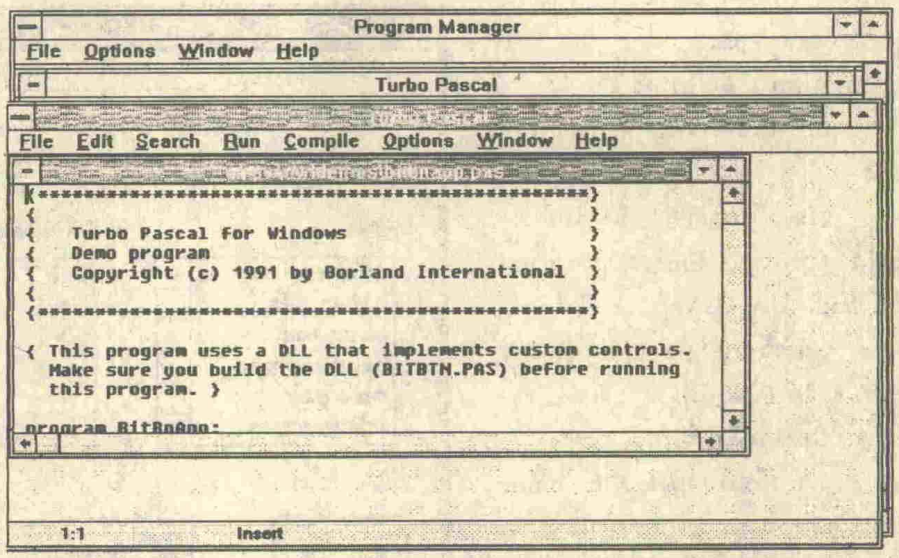
อยู่ 2 หน้าต่าง และมีหน้าต่างดังนี้

– โปรแกรมแมนเนเจอร์ อยู่ล่างสุดและไม่แอคตีฟ

– เทอร์โบปาสคาล เป็นหน้าต่างเอกสารอยู่ถัดขึ้นมาและไม่แอคตีฟ

– ไอดีอี คือ หน้าต่างที่ใช้ชื่อ TURBO PASCAL เช่นกัน ตามรูปทับหน้าต่างเทอร์โบปาสคาลอยู่ ซึ่งบอร์แลนด์เรียกว่า ไอดีอี (IDE : Integrated Development Environment) คือที่ผมเคยเรียกว่าเทอร์โบและที่เคยใช้ในเทอร์โบปาสคาลรุ่นก่อน แต่ตอนนี้อยู่ในรูปของหน้าต่าง

– อีดิต ที่เห็นอยู่ชั้นบนสุด ซึ่งในเทอร์โบปาสคาลก่อนรุ่น 6.0 หน้าต่างอีดิตจะรวมอยู่ในไอดีอี ในรุ่นนี้แยกเป็นหน้าต่างต่างหาก และเปิดใช้ได้ถึง 32 บาน หากมีหน่วยความจำเพียงพอ ตามรูปแสดงโปรแกรมของไฟล์ calc.pas



รูปที่ 1 แสดงภาพบนจอภาพหลังจากรันเทอร์โบปาสคาล

ปรับแต่ง

สำหรับผู้ที่เคยใช้เทอร์โบปาสคาลรุ่นเก่ามา และไม่คุ้นเคยกับการใช้คีย์ในการใช้งานเอดิเตอร์ของวินโดว์ ขอให้ดำเนินการตามนี้ครับ

1. เลือกรายการ Options/Preferences
2. เลือกกรอบ Command Set (tab)
3. เลือกรายการ Alternate (ลูกศร)
4. เลือก OK (Tab ; Enter)

จะเป็นการให้ใช้คีย์ตามแบบของเทอร์โบปาสคาล ซึ่งผมได้ดำเนินการตามนี้ และจะกล่าวตามนี้ด้วยครับ

เตรียมไดเรคทอรี

ในการทำงานต่อไปจะเป็นการป้อนโปรแกรมและเก็บไว้ในไฟล์ .PAS ซึ่งไม่ควรเก็บไว้ในไดเรคทอรีของเทอร์โบปาสคาล คือ TPW ควรเก็บแยกต่างหาก ผมให้เก็บไว้ในไดเรคทอรี TPW__PAS และให้เป็นไดเรคทอรีใหญ่คือ อยู่นอกไดเรคทอรีทั้งปวง ซึ่งไม่ว่าจะรันเทอร์โบปาสคาลก่อนหรือไม่จะดำเนินการได้เหมือนกัน คือ

1. เปิดใช้ Program Manager (หากปิดอยู่)
2. เปิดใช้ Main
3. เปิดใช้ File Manager
4. เลือก C:\ คือ root directory (คีย์ลูกศร) ปกติจะเลือก C:\ อยู่แล้ว
5. เลือกรายการ File/Create directory ของไฟล์แมนเนเจอร์ จะแสดงไดอะล็อกบ็อกซ์ ให้ป้อนชื่อไดเรคทอรี
6. ป้อนชื่อไดเรคทอรี TPW__PAS ที่ Name
7. เลือก OK (Enter)

จะปรากฏชื่อของไดเรคทอรีนี้ในโครงสร้างไดเรคทอรี หากไม่เห็นให้ขยายหน้าต่างหรือสครอล (Scroll) ดู สำหรับหน้าต่างไฟล์แมนเนเจอร์เสร็จแล้วให้มินิไมซ์ไว้เผื่อต้องการใช้งานอีก โดยเลือกรายการ Control/Minimize เสร็จแล้วหากหน้าต่างไอดีอีถูกซ้อนทับอยู่ให้เปิดขึ้นมาไว้ชั้นบน (Alt + Esc)

เปิดไฟล์

ในการป้อนโปรแกรมควรจะเปิดไฟล์ก่อน ถึงแม้จะยังไม่มีไฟล์นั้นในดิสก์ ซึ่งจะดำเนินการได้โดย

1. เลือกรายการ File/Open ของไอดีอี ซึ่งอาจใช้คีย์ด่วน F3 ได้เช่นเคย จะแสดงไดอะล็อกบ็อกซ์ ดังรูปที่ 2

จะเห็นว่าไดเรคทอรีปัจจุบันคือ

C: \ tpw \ owldemos จะต้องเปลี่ยนเป็น C: \ tpw__pas

2. เปลี่ยนไดเรคทอรี

2.1 เลือกกรอบ Directories (tab)

2.2 เลือก [..] คือ ไดเรคทอรีพ่อ (ลูกศร ; Enter) และทำซ้ำจนได้ Path เป็น C: \

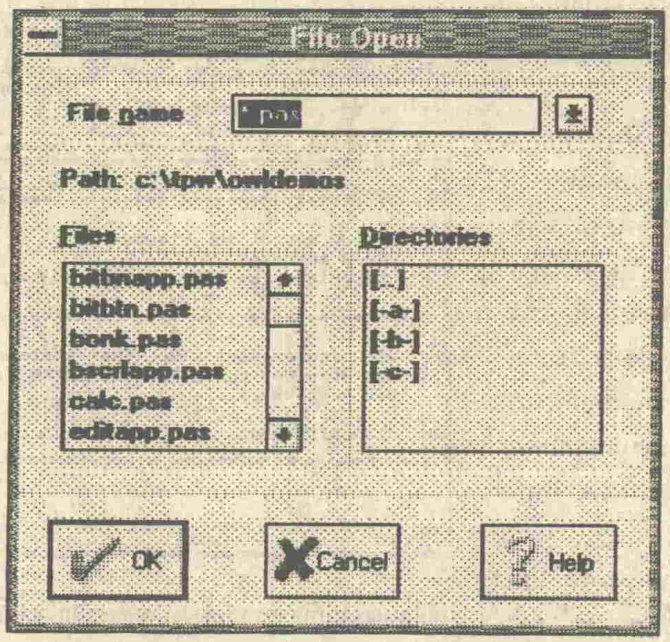
2.3 เลือก [tpw__pas] (ลูกศร ; Enter) จะได้ Path เป็น C: \ tpw__pas

3. ป้อนชื่อไฟล์

3.1 เลือกกรอบ File name (Tab)

3.2 ป้อนชื่อไฟล์ที่ต้องการเปิด หรือเลือกจากกรอบ Files

4. เลือก OK (Enter)



รูปที่ 2 ไดอะล็อกบ็อกซ์ของ File Open ไดเรคทอรีพ่อ คือ ไดเรคทอรีที่อยู่เหนือไดเรคทอรี a: \

เตรียมหน้าต่าง

ในการทำงานกับเทอร์โบปาสคาลต่อไป ควรจะดำเนินการดังนี้ คือ

- มินิไมซ์ Program Manager
- มินิไมซ์ File Manager (หากได้เปิดไว้)
- ปิดหน้าต่าง Main ด้วย Close
- ปิดหน้าต่างเทอร์โบปาสคาลด้วย Close
- แมกซิไมซ์ หน้าต่างไอดีอี

สำหรับหน้าต่างอีดิตให้ปรับขนาดตามความต้องการใช้งาน หากป้อนโปรแกรมยาวควรแมกซิไมซ์ และเมื่อเปิดหน้าต่างอีดิตหลายบาน สามารถจัดระเบียบได้โดย

- เลือกรายการ Window/Tile ซึ่งจะจัดเรียงหน้าต่างในแบบเรียงติดกัน แบ่งเนื้อที่บนบานหน้าต่างของไอดีอี
- หรือเลือกรายการ Window/Cascode ซึ่งจะเรียงหน้าต่างในแบบเหลื่อมทับกัน

```
PROGRAM  MyFirstInTPW;
USES  WinCrt;
BEGIN
  WriteLn('This is my first Program in');
  WriteLn('TURBO PASCAL FOR WINDOWS');
end.
```

รูปที่ 3 โปรแกรม MyFirst

โปรแกรมแรก

เพื่อทำความคุ้นเคยกับการใช้งานเทอร์โบฯ และเป็นตัวอย่างในการป้อนคอมพายล์ และรัน ขอให้ดำเนินการดังนี้

1. เปิดไฟล์

เปิดไฟล์ MYFIRST.PAS ตามวิธีที่กล่าวมา จะเปิดหน้าต่าง myfirst.pas ให้อีก 1 หน้าต่าง โดยที่หน้าต่าง calc.pas เดิมยังคงอยู่

2. ป้อนโปรแกรม

ขอให้ป้อนโปรแกรมตามรูปที่ 3 โดยที่ในหน้าต่างอีดิตจะมีเคอร์เซอร์รูปที่ตัว I กะพริบ นอกเหนือไปจากพอยน์เตอร์ของเมาส์ ขอให้ดำเนินการได้เช่นเดียวกับในเทอร์โบปาสคาลรุ่นก่อน

ในส่วนของโปรแกรมที่แปลกใหม่มาคือให้ใช้ (uses) WinCrt เพื่อให้ใช้คำสั่ง WriteLn (รวมทั้ง ReadLn) ในกราฟิกโหมดของวินโดวส์ได้

3. คอมพายล์

- เลือกรายการ Compile/Compile หากคอมพายล์ไม่ผ่านจะแสดงข้อความถึงข้อผิดพลาดที่สเตตัสบาร์ (บรรทัดล่างสุด) และแสดงเคอร์เซอร์ ณ ตำแหน่งที่ผิด การคอมพายล์จะคอมพายล์เป็นไฟล์ .EXE

4. เก็บลงไฟล์

เมื่อคอมพายล์ผ่านแล้วควรเก็บลงไฟล์ โดย

– เลือกรายการ File/Save ซึ่งอาจใช้คีย์ F2 ได้เช่นเคย

5. รันโปรแกรม

– เลือกรายการ Run/Run จะแสดงผลการรันในหน้าต่างที่เปิดใหม่อีก 1 หน้าต่าง เป็นหน้าต่างงานของวินโดวส์ เมื่อทราบผลแล้วให้ปิดด้วย Close

การลอกข้อความ

สามารถจะลอกข้อความจากหน้าต่างหนึ่งไปให้แก่อีกหน้าต่างหนึ่งได้ ขอให้เปิดไฟล์ SECOND.PAS ซึ่งจะเป็นการเปิดหน้าต่างนี้ด้วย โดยจะลอกข้อความจากหน้าต่าง myfirst.pas มายังหน้าต่างหลังนี้ ซึ่งจะดำเนินการได้ดังนี้

1. เปิดใช้งาน myfirst.pas
2. หมายกลุ่มข้อความ ซึ่งอาจจะใช้คีย์ตามแบบเดิม (Ctrl + K,B) หรือดำเนินการตามที่จะกล่าวในหัวข้อ การหมายกลุ่มข้อความ ขอให้หมายกลุ่มข้อความตั้งแต่ PROGRAM ถึง BEGIN
3. เลือกรายการ Edit/Copy
4. เปิดใช้งาน second. pas
5. เลือกรายการ Edit/Paste จะปรากฏข้อความที่หมายกลุ่มไว้ในหน้าต่างใหม่นี้

การหมายกลุ่มข้อความ

การหมายกลุ่มข้อความเพื่อจะดำเนินการต่อไป สามารถดำเนินการได้ 3 วิธี คือ

ก. ใช้เม้าส์

คลิ๊กค้างไว้ที่ต้นกลุ่มข้อความแล้วเลื่อนพอยน์เตอร์ จะแสดงไฮไลต์ตามไปด้วย ไปท้ายกลุ่มข้อความปล่อยมือ หากท้ายกลุ่มข้อความไม่ได้แสดงในหน้าต่างให้เลื่อนพอยน์เตอร์ออกไปนอกกรอบแล้วปล่อยมือ หากต้องการยกเลิกบางส่วนให้เลื่อนพอยน์เตอร์ย้อนหลังก่อนปล่อยมือ ถ้าปล่อยมือแล้วคลิ๊กใหม่จะเป็นการยกเลิกทั้งหมด

ข. ใช้คีย์เดิม

กดคีย์ Ctrl + K,B ที่ต้นกลุ่มข้อความ แล้วใช้คีย์ในกลุ่มคีย์ลูกศร (เช่นเดียวกับในเทอร์โบปาสคาลรุ่นก่อน) เลื่อนไฮไลต์ไปท้ายกลุ่มข้อความ (ไม่ต้องกดคีย์ Ctrl+K,K)

ค. ใช้คีย์ใหม่

ใช้คีย์ shift+ คีย์ในกลุ่มคีย์ลูกศร เช่นเดียวกับที่กล่าวในข้อ ข. เลื่อนไฮไลต์จากต้นกลุ่มข้อความไปท้ายกลุ่มข้อความ

เตรียมใช้ดีบักเกอร์

การเขียนโปรแกรมไม่ให้มีที่ผิดเลยเป็นไปได้ยาก บอร์แลนด์จึงได้จัดโปรแกรมสำหรับตรวจหาที่ผิดคือ ดีบักเกอร์ เพื่อใช้กับเทอร์โบปาสคาลมาหลายรุ่นแล้ว สำหรับรุ่นที่ใช้กับวินโดวส์ให้ชื่อว่า TDW (Turbo Debugger for Windows)

สำหรับท่านที่ใช้รุ่นก่อนมาแล้วคงอืดอัด เพราะต้องเตรียมการจะใช้ตรง ๆ อย่างก่อนไม่ได้ ทั้งนี้เพราะ TDW ไม่ได้อยู่บนวินโดวส์คงรันในเท็กซ์โหมด และการรัน TDW รันได้หลายแบบ ผมจะกล่าวถึงเฉพาะการรันจากเทอร์โบปาสคาลคือ จากที่เราป้อนโปรแกรมโดยที่ TDW ให้เราเตรียมการดังนี้

– ให้ไฟล์ .PAS และ .EXE อยู่ในไดเรคทอรีเดียวกัน จากการดำเนินการที่ผ่านมาจะอยู่ในไดเรคทอรีเดียวกันอยู่แล้ว

– ให้ข้อมูลเกี่ยวกับการหาข้อผิดพลาดไว้ในไฟล์ .EXE ซึ่งจะทำได้โดย

1. เลือกรายการ Option/Linker
2. เซต Debug info in Exe ให้เป็นเครื่องหมายถูก (Tab ; Space)
3. เลือก OK (Enter)
4. เลือกรายการ Option/Compiler
5. เซต Debug information ในกรอบ Options ให้เป็นบวก (Tab ; ลูกศร, Space)
6. เซต Local symbol ในกรอบเดียวกัน (ลูกศร, Space)
7. เลือกรายการ OK (Enter)

รายการดังกล่าวมานี้อาจเซตไว้แล้ว จึงเพียงแต่ตรวจสอบดู ตอนนี้ก็พร้อมที่จะใช้ TDW ขอให้ป้อนโปรแกรม

```
PROGRAM SecondProg;
USES  WinCrt;
VAR  a,b : Integer;  r : Real;

BEGIN
  repeat
    Write('Enter two number: ');
    ReadLn(a,b);
    r := a / b;
    WriteLn('The ratio is ',r:3:2);
  until  b = 0;
END.
```

รูปที่ 4 โปรแกรม Second

ตามรูปที่ 4 ไว้ในไฟล์ SCOND.PAS (ที่ดำเนินการค้างอยู่) โดยแก้ชื่อโปรแกรม แทรกบรรทัด VAR และป้อนต่อ

ตรวจหาที่ผิด

โปรแกรมนี้มีที่ผิดตรงที่ ให้ผู้ใช้ป้อนเลข 2 จำนวนนำมาหารกัน หากผู้ใช้ป้อนเลขจำนวนหลังเป็น 0 จะเกิด run time error เนื่องจากการหารด้วย 0 ซึ่งในโปรแกรมจะต้องป้องกันไว้ แต่ไม่ได้ดำเนินการ ขอให้ทดลองรันโปรแกรมดูก่อนครับ

เมื่อโปรแกรมมีที่ผิด (ซึ่งต้องถือว่าเราไม่ทราบว่าผิดตรงจุดใด) ก็ต้องตรวจหาที่ผิดด้วยดีบักเกอร์ ซึ่งจะดำเนินการได้โดยรัน TDW คือ

– เลือกรายการ Run/Debugger

จะเป็นการโหลดและรัน TDW บนดอส แสดงหน้าต่างของ TDW เอง (ไม่ใช่ของวินโดวส์) การใช้จึงเป็นไปตามแบบของเทอร์โบปาสคาล เช่น การเลือกรายการจะเป็น Alt+

ในหน้าต่างจะแสดงโปรแกรม second. pas ให้ตรวจแก้ มีดัชนี (ผมไม่ทราบว่าจะเรียกอย่างไรดี) เป็นรูปสามเหลี่ยมอยู่ตรง BEGIN (แทนไฮไลต์ที่เป็นบาร์เดิม) จึงมีของอยู่ 3 สิ่งทำหน้าที่ต่างกัน คือ

– พอยน์เตอร์ของเม้าส์ สำหรับการใช้งานเม้าส์

– เคอร์เซอร์ สำหรับสร้างเงื่อนไขในการรัน

– ดัชนีสำหรับแสดงว่ารันโปรแกรมถึงที่ใด

การตรวจหาที่ผิดคงคล้ายกับเทอร์โบปาสคาลรุ่นเก่า ขอให้ลองตรวจหาที่ผิดตามนี้ครับ

1. เปิดหน้าต่าง Watch ด้วย F6 แล้วป้อน a จะแสดงหน้าต่างให้เราป้อนต่อแต่ให้กด Enter เพราะชื่อตัวแปรมีเพียงอักขระเดียว และให้ทำเช่นนี้กับ b และ a/b (ป้อนติดต่อกัน) แล้วกด F6 กลับไปหน้าต่างดีบัก

2. ทำซิงเกิลสเตป ด้วยคีย์ F7 หรือ F8 จนผ่านคำสั่ง ReadLn ซึ่งจะต้องป้อนเลข 2 จำนวน ในเที่ยวแรกให้ป้อน b มีค่าที่ไม่ใช่ 0 เพื่อดูค่า a/b แต่ผมจะอธิบายในลักษณะที่ป้อน b เป็น 0 ซึ่งจะได้ว่า a/b เป็น +INF แสดงว่ามีสิ่งผิดปกติเกิดขึ้น

3. เมื่อซิงเกิลสเตปผ่านประโยคให้ค่าแก่ r จะเกิด run time error แสดงในบ็อกซ์ให้กด Enter รับทราบ กลับมาที่ดีบักเกอร์ แจ้งโค้ดของการยุติการรัน

4. ให้กด Enter รับทราบ บนจอภาพจะแสดงหน้าต่างซีพียู ให้เก็บหน้าต่างนี้ด้วย Alt+F3 กลับไปหน้าต่างดีบัก

5. ให้ยุติการรัน TDW ด้วย Alt+X หรือเลิกรายการ File/Quit กลับไปวินโดวส์

แก้ที่ผิด

เมื่อทราบที่ผิดก็ต้องหาทางป้องกัน ดังโปรแกรมใหม่ได้ตามรูปที่ 5 จะเห็นว่าไม่สามารถแก้ที่ผิดใน TDW ได้ ต้องมาทำในเทอร์โบปาสคาล เมื่อแก้ที่ผิดแล้วปกติต้องหาที่ผิดอื่นต่อไป ซึ่งในตอนนี้จะไม่คอมพายล์ก็ได้แต่ขอให้เก็บลงไฟล์ แล้วรัน TDW ใหม่ (เป็นความไม่สะดวกอีกครับ)

ยุติการรันเทอร์โบปาสคาล

สำหรับเทอร์โบปาสคาลคงมีเท่านี้ก่อนครับ ขอให้ยุติการรัน โดย

– เลือกรายการ File/Exit ซึ่งใช้คีย์ด่วน Alt+x ได้เช่นกัน

ทดลองรัน .EXE

เพื่อทำความคุ้นเคยและทำความเข้าใจกับแอพพลิเคชันที่จะอยู่บนหรือไม่บนวินโดวส์ จะทดลองรันโปรแกรมที่เพิ่งป้อนใหม่นี้ ขอให้ดำเนินการตามนี้ครับ

1. เปิดใช้ Main
2. เปิดใช้ File Manager
3. ให้แสดงไดเรคทอรีของ TPW __PAS (คลิ๊กซ้อนที่ชื่อ : ถูกศร ; Enter) Enter) จะเห็นไฟล์ .EXE สองไฟล์ คือ MYFIRST.EXE กับ SECOND.

```
PROGRAM SecondProg;
USES  WinCrt;
VAR  a,b : Integer;  r : Real;

BEGIN
  repeat
    Write('Enter two number: ');
    ReadLn(a,b);
    if  b <> 0  then
      begin  r := a / b;
             WriteLn('The ratio is ',r:3:2)  end
    else  WriteLn('The ratio is undefied');
  until  b = 0;
END.
```

รูปที่ 5 โปรแกรม Second หลังจากที่แก้ที่ผิดแล้ว

EXE

4. รันโปรแกรม SECOND. EXE (คลิ๊กซ้อนที่ชื่อ : ลูกศร ; Enter)

จะแสดงหน้าต่างของโปรแกรมนี้บนจอภาพเช่นเดียวกับที่เห็นเมื่อรันในเทอร์โบปาสคาล โปรแกรมประเภทนี้จะถือว่าอยู่บนวินโดวส์ก็ไม่ใช่ทีเดียวนัก แต่จะว่าไม่อยู่ก็ไม่ได้เพราะรันบนดอสไม่ได้

ขอให้มินิไมซ์หน้าต่างนี้แล้วรันโปรแกรมนี้ใหม่ จะเห็นว่าวินโดวส์จัดหน้าต่างให้ใหม่ แต่ถ้าลองรัน TPW. EXE (ซึ่งเป็นแอพพลิเคชันบนวินโดวส์แท้ ๆ) ด้วยวิธีนี้ จะรันได้เพียงครั้งเดียว ก็ต้องไปทำความเข้าใจกันในฉบับต่อ ๆ ไปครับ สำหรับไฟล์ .EXE ทั้งสองนี้ ผมคิดว่าไม่จำเป็นต้องเก็บไว้

สองจอภาพ

การตรวจหาที่ผิดในโปรแกรมหากใช้ 2 จอภาพ จะทำได้สะดวกและเร็วกว่า 1 จอภาพมาก เพราะสามารถที่จะดูขั้นตอนในการรันกับค่าในตัวแปรและอื่น ๆ อีกที่จอภาพหนึ่ง โดยดูผลการรันของโปรแกรมที่นำมาตรวจหาที่ผิดที่อีกจอภาพหนึ่ง

ขึ้นกับฮาร์ดแวร์

โดยหลักการแล้วการ์ด VGA ทุกรุ่นสามารถใช้ร่วมกับการ์ดเฮอร์คิวลิสหรือ CGA ได้ แต่ของจริงไม่เป็นเช่นนั้น บางการ์ดทำไม่ได้ ผมคิดว่าจำเป็นต้องบอกชื่อที่ผมติดตั้งอยู่ใช้การ์ด VGA ของ Ultra VGA ที่ใช้ชิพ ET4000 (คิดว่าการ์ดนี้ใช้ชิพนี้คงใช้ได้ทั้งหมด) โดยติดตั้งร่วมกับการ์ดเฮอร์คิวลิส

เมื่อไม่แน่ใจก็ต้องทดสอบครับ คือ หลังจากติดตั้งแล้วให้ใช้คำสั่งดอสว่า mode mono เครื่องจะทำงานบนจอ

โมโนโครม ให้ไดเรคทอรีดูว่ามีสิ่งผิดปกติบนจอ VGA หรือไม่ แล้วใช้คำสั่ง mode co80 กลับไปจอ VGA ไดเรคทอรีดูว่ามีสิ่งผิดปกติบนจอโมโนโครมหรือไม่ ถ้าไม่มีก็ใช้ได้ครับ

ทางด้านซอฟต์แวร์

บอร์แลนด์ได้เตรียมการไว้ให้แล้ว เป็นโปรแกรมในไฟล์ TDWINST. EXT อยู่ในไดเรคทอรี C:\TPW\UTILS ซึ่งจะรันบนดอสหรือบนวินโดว์ก็ได้ ขอให้ดำเนินการดังนี้ครับ

1. รัน tdwinst แสดงเมนูหลัก

2. เลือกรายการ Display จะแสดงไดอะล็อกบ็อกซ์ให้ปรับแต่งค่าต่าง ๆ

3. เลือกกรอบ User screen updating ด้วยคีย์ Tab แล้วเลือกรายการ Other display ด้วยคีย์ลูกศร หากใช้เม้าส์ให้คลิ๊กที่รายการนี้โดยตรง

4. เลือก OK (Enter) กลับไปเมนูหลัก

5. เลือก Save แสดงรายการให้เลือก 2 รายการ ของผมเลือกรายการ Modify tdw.exe

6. จบด้วย Quit

จะเป็นการติดตั้งจอภาพ 2 จอกับ tdw ทดสอบดูได้ โดยรัน tdwinst อีกครั้ง จะเปลี่ยนไปใช้จอโมโนโครมเช่นเดียวกับเมื่อใช้ tdw ในเทอร์โบปาสคาล tdw จะใช้จอโมโนโครมและแสดงผลการรันของโปรแกรมเราบนวินโดว์บนจอ VGA ขอให้ทดลองใช้งานดูครับ

ทบทวนโอโอพี

การเขียนโปรแกรมบนวินโดวส์ด้วยเทอร์โบปาสคาลที่จะกล่าวต่อไปจะต้องเขียนในแบบโอโอพี (OOP : Object-Oriented Programming) ท่านที่ไม่คุ้นเคยกับโอโอพีไม่ต้องตกใจครับ เพราะจะใช้เพียงผิวเผินเท่านั้น และในรูปแบบซ้ำ ๆ กัน แต่เพื่อประกอบความเข้าใจโดยที่ผมว่างเว้นในเรื่องนี้ไปนาน จึงขอนำมาทบทวนโดยสังเขป แต่คาดว่าเพียงพอ

การเขียนโปรแกรมในแบบโอโอพี คือ การเขียนโปรแกรมโดยมุ่งที่ออปเจ็กต์ ออปเจ็กต์คือ สิ่งที่สามารถสร้างแบบจำลองได้ด้วยโปรแกรมคอมพิวเตอร์ โดยสร้างเป็นสกุล (Class) แล้วกำหนดอินสแตนซ์ (instance) คือ ตัวแปรของสกุล อินสแตนซ์หนึ่ง ๆ นี้คือออปเจ็กต์หนึ่ง ๆ แต่จะเป็นออปเจ็กต์ที่แท้จริงต่อเมื่อให้ค่าแล้ว ดูตัวอย่างดีกว่าครับ

โปรแกรมตามแบบ

หากผมจะเขียนโปรแกรมแสดงภาพจุดบนจอภาพและให้ย้ายที่ได้ จะเขียนเป็นโปรแกรมตามแบบของภาษาปาสคาลธรรมดาได้ตามรูปที่ 6 คือ

– ได้กำหนดตัวแปรไว้สำหรับจุด 3 จุด แต่ใช้ในการเขียนโปรแกรมเพียง

จุดเดียว

– ตอนต้นให้ PrepGraph เตรียมการทางด้านกราฟิกให้ (ซึ่งของดแสดงโปรแกรมย่อย)

– ให้ค่าแก่จุดซึ่งสมมติเป็นจุด a แล้วให้แสดงตัวด้วย Show คือ เขียนภาพจุดบนจอภาพ (ไม่ได้แสดงโปรแกรมย่อย)

– แล้วให้ย้ายตำแหน่งด้วย MoveTo ซึ่งใน MoveTo จะลบภาพจุดเดิมด้วย Hide (ไม่ได้แสดงโปรแกรมย่อยอีกเช่นกัน) และแสดงภาพจุด ณ ตำแหน่งใหม่ด้วย Show

จะเห็นว่าในการเขียนโปรแกรมจะต้องจดหรือจำชื่อตัวแปรไว้ตลอดเวลา ซึ่งถ้าต้องการเขียนโปรแกรมสำหรับจุด 3 จุด ก็ต้องจำชื่อตัวแปร 6 ตัว ดูแล้วก็เป็นสิ่งเล็กน้อย แต่สิ่งเล็ก ๆ น้อย ๆ เช่นนี้ เมื่อมากเข้าจะทำให้การเขียนโปรแกรมโดยส่วนรวมล่าช้า จึงได้คิดเขียนโปรแกรมในแบบโอโอพีขึ้นครับ ตามที่จะกล่าวต่อไป

```
PROGRAM  Point;
VAR      xa,ya,xb,yb,xc,yc : Integer;

PROCEDURE  MoveTo(var ox,oy : Integer; nx,ny : Integer);
BEGIN
  Hide(ox,oy);  ox := nx;  oy := ny;  Show(ox,oy);
END;


BEGIN
  PrepGraph;
  xa := 50;  ya := 100;  Show(xa,ya);
  MoveTo(xa,ya,100,200);
END.
```

รูปที่ 6 โปรแกรมตามแบบภาษาปาสคาลมาตรฐาน

```
PROGRAM  ObjectPoint;

TYPE
  Point = OBJECT
    x,y : Integer;
  procedure  Init(xx,yy : Integer);
  procedure  Show;
  procedure  Hide;
  procedure  MoveTo(nx,ny : Integer);
          END;


PROCEDURE  Point.Init(xx,yy : Integer);
BEGIN    x := xx;  y := yy    END;

PROCEDURE  Point.MoveTo(nx,ny : Integer);
BEGIN  Hide;  x := nx;  y := ny;  Show  END;


VAR       pntA,pntB,pntC : Point;
BEGIN
  pntA.Init(50,100);  MoveTo(100,200);
END.
```

รูปที่ 7 โปรแกรมในแบบโอโอพี

เอนแคปซูลเลชัน (Encapsulation)

จากที่กล่าวมาแล้วได้นำมาใช้เป็นหลักการข้อหนึ่งของการเขียนโปรแกรมในแบบโอโอพี คือ ให้สามารถรวมข้อมูล (ตัวแปร) กับโปรแกรมย่อยที่จะดำเนินการกับมันไว้ในที่แห่งเดียวกันได้ เรียกว่า เอนแคปซูล ดังตัวอย่างในรูปที่ 7 เป็นโปรแกรมที่ดำเนินการเช่นเดียวกับในรูปที่ 6 แต่เขียนในแบบโอโอพี กล่าวคือ (7)

– ได้กำหนดสกุล Point ตามข้อกำหนดของเทอร์โบปาสคาลคือ เป็นแบบข้อมูล OBJECT คล้ายกับแบบแบบข้อมูล RECORD ต่างกันตรงมีฟิลด์โปรแกรมย่อยเพิ่มขึ้น

– สำหรับโปรแกรมย่อยในสกุลมีชื่อเรียกใหม่ว่า กิจกรรม (meltod) ส่วนข้อมูลคือตัวแปรเรียกว่า รายละเอียด (attribute)

– โดยที่กิจกรรมต้องดำเนินการกับรายละเอียดเท่านั้น (ห้ามดำเนินการกับข้อมูลนอกสกุล)

– จะเห็นว่าไม่ได้ผ่านค่าให้ Hide และ Show เพราะว่าต้องดำเนินการกับรายละเอียดอยู่แล้ว

– เมื่อกำหนดสกุลแล้วให้นำไปเขียนโปรแกรมย่อยของแต่ละกิจกรรมดังที่เขียนให้แก่ Init และ MoveTo (และขอแสดงเฉพาะสองกิจกรรมนี้เท่านั้น)

– ที่ต้องกำหนดกิจกรรม Init (ซึ่งจะใช้ชื่อใดก็ได้) เพราะตามหลัก

ของการเขียนโปรแกรมในแบบโอโอพี ห้ามโปรแกรมนอกสกุลให้ค่าแก่ตัวแปรในสกุล เป็นการปกป้อง

– ได้กำหนดอินสแตนซ์ของสกุล Point ไว้ 3 ตัวคือ pntA, pntB และ pntC เป็นจุด 3 จุด แต่นำไปใช้ในโปรแกรมเพียงตัวเดียว

– อินสแตนซ์แต่ละตัวนี้จะมีตัวแปร X และ Y เป็นของตนเอง (เช่นเดียวกับตัวแปรเรคอร์ด)

– ในโปรแกรมหลักได้ให้ดำเนินการเช่นเดียวกับโปรแกรมแรก

จะเห็นว่าไม่ต้องจำชื่อตัวแปรที่เป็นรายละเอียดอีกต่อไป และรายละเอียดรวมทั้งกิจกรรมอยู่ในที่ปกป้องโปรแกรมส่วนอื่นจะเข้าถึงโดยตรงไม่ได้ ต้องผ่านอินสแตนซ์ของสกุลดังที่เห็นในโปรแกรมหลัก และสำหรับรายละเอียดต้องผ่านกิจกรรม เช่นตามตัวอย่างนี้ คือ Init และ MoveTo ทำให้ประกันได้ว่าจะไม่มีการเปลี่ยนค่าโดยไม่ตั้งใจ

อินเฮอริต (Inheritance)

ต่อมาหากผมต้องการสร้างสกุลวงกลมคือ Circle ก็สามารถดำเนินการได้เช่นเดียวกับที่สร้างสกุล Point แต่โอโอพีให้ทำได้ดีกว่าตามหลักของการอินเฮอริตคือ การสืบสกุล ซึ่งหมายรวมถึง การสืบมรดกด้วย ดังโปรแกรมในรูปที่ 8 คือ

– ให้สกุล Circle เป็นผู้สืบสกุลของ Point

– ทำให้สามารถใช้ทั้งรายละเอียดและกิจกรรมของบรรพบุรุษได้ ดังที่จะเห็นว่าไม่ได้กำหนดตัวแปรสำหรับเก็บค่าโคออร์ดิเนตของจุดศูนย์กลางของวงกลม เพราะมีอยู่แล้วใน Point

```
TYPE
   Circle   : OBJECT(Point)
     r      : Integer;
   constructor Init(xx,yy : Integer);
   procedure  Show; virtual;
   procedure  Hide; virtual;
          END;
```

รูปที่ 8 แสดงการสืบสกุล

– สำหรับกิจกรรมก็เช่นกัน จะใช้ MoveTo ของ Point แต่มีเรื่องอื่นด้วย ดังจะนำไปกล่าวในหัวข้อต่อไป

โพลิมอร์พิซึม (Polymorphism)

หากดูโปรแกรมในรูปที่ 6 และ 7 เทียบกันจะเห็นว่ามีกิจกรรม Hide กับ Show เหมือนกัน กิจกรรมทั้งสองคู่นี้แตกต่างกันในทางปฏิบัติคือ Hide ของ Point จะลบจุด 1 จุด ส่วน Hide ของ Circle จะลบวงกลม 1 วง เช่นกัน Show จะเป็นการวาดจุดและวงกลมตามลำดับ

และถ้าดูในโปรแกรมของ MoveTo (รูปที่ 7) จะเห็นว่ามีการเรียกใช้กิจกรรมทั้งสองนี้ นั่นคือ ถ้ากำหนดกิจกรรม MoveTo ใน Circle จะสามารถลอกโปรแกรมของ MoveTo ใน Point มาใช้ได้โดยตรง ที่ไม่กำหนดเพราะต้องการใช้ MoveTo ของ Point ซึ่งถ้าหลักการของโอโอพีมีเพียง 2 เรื่องดังกล่าวมาจะทำไม่ได้ แต่สามารถกระทำได้ตามหลักการข้อที่สามคือ โพลิมอร์-พิซึม หมายถึง เลือกการปฏิบัติ โดยเทอร์โบปาสคาลกำหนดไว้ดังนี้ คือ

– ให้กิจกรรมที่จะถูกเลือกปฏิบัติเป็นเวอร์ชวล (Virtual) ตามตัวอย่างที่ยกมา คือ Hide กับ Show ดังที่เห็นในรูปที่ 7 และต้องย้อนกลับไปแก้ของ Point ให้เป็นเวอร์ชวลด้วย

– จัดกิจกรรมหนึ่ง ซึ่งจะใช้เป็นกิจกรรมแรกเป็นคอนสตรัคเตอร์ โดยใช้คำว่า Constructor หน้าชื่อกิจกรรม (แทน procedure) ซึ่งเรามี Init อยู่แล้วจึงให้เป็นคอนสตรัคเตอร์ ดังที่ใช้ในรูปที่ 8 และต้องย้อนกลับไปแก้ให้ Init ของ Point เป็นคอนสตรัคเตอร์เช่นกัน

– การกำหนดกิจกรรมเป็นคอน-สตรัคเตอร์ก็เพื่อการสร้างตาราง VMT (Virtual Method Table)

หลังจากดำเนินการเช่นนี้แล้ว เมื่อใช้ MoveTo กับอินสแตนซ์ของสกุล Point จะเป็นการใช้ Hide กับ Show ของ Point แต่ถ้าใช้ MoveTo กับอินสแตนซ์ของสกุล Circle จะเป็นการใช้ Hide กับ Show ของ Circle ทำให้เกิดการปฏิบัติที่แตกต่างกัน เป็นการเลือกปฏิบัติผ่านทางตาราง VMT

เรื่องราวของโอโอพีโดยสังเขปมีเพียงเท่านี้ครับ หากศึกษาต่อก็จะเป็นรายละเอียดของแต่ละเรื่อง และประโยชน์ของโพลิมอร์พิซึมที่กล่าวมานี้เป็นเพียงข้อเดียว อีกข้อหนึ่งดังจะเห็นในของจริงที่จะกล่าวต่อไป

ตาราง VMT กับ DMT

เพื่อให้ทำงานในแบบเวอร์ชวลได้ เทอร์โบปาสคาลได้ออกแบบโดย

ทำเป็นตาราง VMT (Virtual Method Table) เพื่อเก็บพอยน์เตอร์ไปยังโปรแกรมย่อยของแต่ละกิจกรรมที่เป็นเวอร์ชวล

เช่น สกุล Alpha มีกิจกรรม One ถึง Ten และมีกิจกรรม Five ถึง Ten เป็นเวอร์ชวล ก็จะมีพอยน์เตอร์ไปยังโปรแกรมย่อยของกิจกรรม Five ถึง Ten รวม 6 พอยน์เตอร์

ต่อมาสร้างสกุล Bravo เป็นผู้สืบสกุลของ Alpha สกุล Bravo จะมีตาราง VMT เช่นเดียวกับ Alpha และลอกค่าในตารางมาให้ด้วย ซึ่งถ้าในสกุล Bravo ได้กำหนดกิจกรรม Six ทับก็จะเปลี่ยนค่าพอยน์เตอร์ของ Six เดิมเป็นของ Six ใหม่ ทำให้สามารถทำงานในแบบเวอร์ชวลได้

วิธีการเช่นนี้ใช้ได้ดีสำหรับสกุลที่มีกิจกรรมเป็นเวอร์ชวลไม่มากนัก แต่ถ้ามีกิจกรรมเป็นเวอร์ชวลมาก เช่น เป็นร้อยจะสิ้นเปลืองหน่วยความจำมาก เข้าใจว่าบอร์แลนด์พบปัญหานี้เมื่อจะเขียนโปรแกรมสำหรับเทอร์โบปาสคาลบนวินโดวส์ ซึ่งต้องกำหนดกิจกรรมเป็นเวอร์ชวลในสกุลต้นกำเนิดมากกว่า 200 กิจกรรม และสกุลนี้มีผู้สืบสกุลในขั้นลูกประมาณ 20 สกุล และยังจะมีขั้นหลานต่อไปอีกทำให้หมดเปลืองหน่วยความจำมาก

จึงได้เพิ่มตารางเวอร์ชวลอีก 1 แบบ เรียกว่า DMT (Dynammic Method Tables) ไม่ต้องสนใจชื่อหรอกครับ ดูที่หลักการจะเข้าใจกว่า วิธีใหม่นี้ให้แต่ละกิจกรรมที่เป็นเวอร์ชวลมีหมายเลขประจำตัว (ผู้เขียนโปรแกรมเป็นผู้ให้) เช่นรูปที่ 9

ในการจัดทำตาราง เช่น เมื่อบรรพบุรุษมีกิจกรรมเป็นเวอร์ชวล 200 กิจกรรม ก็จะมีตารางหมายเลขประจำตัว 200 ช่อง กับตารางพอยน์เตอร์เรียงลำดับตรงกันอีก 200 ช่อง เมื่อกำหนดสกุลลูกสมมติว่าได้กำหนดกิจกรรมที่เป็นเวอร์ชวลซ้ำของบรรพบุรุษ 10 กิจกรรม และกำหนดใหม่อีก 2 กิจกรรม ในสกุลใหม่นี้จะมีตาราง DMT สำหรับเลขประจำตัวกับพอยน์เตอร์เพียง 12 คู่เท่านั้น

```
procedure One ; virtual 10;
procedure Two ; virtual 20;
```

รูปที่ 9

```
procedure WMLBultamDown (var msg : TMessage);
          virtual wm__First + wm__LBultonDown;
```

รูปที่ 10

จะเห็นว่าใช้หน่วยความจำน้อยกว่ากันมากคือ 72 ไบต์ กับ 808 ไบต์ หากใช้เป็นตาราง VMT และจำนวน 10 ที่ยกมาเป็นตัวอย่างเป็นจำนวนค่อนข้างมาก แต่ที่สำคัญคือ ของเทอร์โบปาสคาลเองที่มีสกุลลูกประมาณ 20 สกุลแทบไม่มีการกำหนดกิจกรรมซ้ำ

ในด้านการทำงานโดยหลักการคือ เมื่อมีการเรียกใช้กิจกรรมกับอินสแตนซ์ของสกุลลูก จะนำหมายเลขประจำตัวของกิจกรรมไปค้นหาในตารางหมายเลขประจำตัวของสกุลลูกก่อน หากไม่พบจะค้นในสกุลพ่อ (และต่อ ๆ ไปหากมี) เมื่อพบจะนำค่าพอยน์เตอร์ไปใช้ต่อไป

การดำเนินการเช่นนี้ทำให้ใช้เวลามากกว่าตาราง VMT มาก จึงไม่ควรใช้กับการเขียนโปรแกรมตามปกติ ซึ่งจะมีกิจกรรมที่เป็นเวอร์ชวลไม่มากนัก

DMT กับสื่อความของวินโดวส์

ที่ผมต้องนำมากล่าวถึง เพราะว่าในการเขียนโปรแกรมบนวินโดวส์จะต้องปรับปรุงคือกำหนดกิจกรรมทับกิจกรรมของเทอร์โบปาสคาลที่เป็นแบบ DMT เพื่อสนองตอบการเรียกใช้จากสื่อความของวินโดวส์ (ซึ่งจะกล่าวในฉบับต่อไป) เช่นรูปที่ 10

โดยที่ WMLBultomDown คือชื่อโพรซีเดอร์ที่จะสนองตอบสื่อความ wm__LBultonDown ของวินโดวส์ ซึ่งสื่อความมีมากด้วยกันดังกล่าวมา เทอร์โบปาสคาลจึงทำให้ง่ายขึ้น โดย

- เมื่อจะดำเนินการกับสื่อความใดให้ตั้งชื่อกิจกรรมเช่นเดียวกับชื่อสื่อความ แต่ไม่มีขีดล่าง
- มีอาร์กิวเมนต์เหมือนกันหมดคือ var msg: TMessage
- หมายเลขประจำตัวให้เป็น wm__First + ชื่อสื่อความ

ทั้งนี้ต้องดำเนินการในผู้สืบสกุลของสกุลในเทอร์โบปาสคาลที่จะกล่าวในตอนต่อไป นำมาเรียนให้ทราบไว้ชั้นหนึ่งก่อนครับ

ก่อนจบตอน

ฉบับนี้จึงเป็นแค่เพียงเรื่องราวของเทอร์โบปาสคาล เพื่อเตรียมไว้เขียนโปรแกรมบนวินโดวส์ในตอนต่อไปครับ